Tutorial on statistical power analysis via simulations in R

Andrey Chetverikov

January 28, 2020

```
# this chunk loads the libraries and sets up ggplot theme
library(apastats)
library(ggplot2)
library(data.table)
library(Hmisc)
library(patchwork)
library(MASS)
library(knitr)
library(ez)
library(pwr)

# packages from github
library(apastats) # devtools::install_github('achetverikov/apastats',subdir='apastats')
library(patchwork) # devtools::install_github("thomasp85/patchwork")
library(Superpower) # devtools::install_github("arcaldwell49/Superpower")

default_font <- 'sans'
default_font_size <- 12
default_font_size_mm <- default_font_size/ggplot2:::.pt

default_theme<-theme_light(base_size = default_font_size, base_family = default_font)+theme(
  axis.line=element_line(size=I(0.5)),
  axis.ticks= element_line(size=I(0.25), colour = 'gray'),
  axis.line.x=element_line(),
  axis.line.y=element_line(),
  panel.grid=element_line(colour = 'gray',size = I(0.5)),
  panel.grid.minor = element_blank(),
  legend.title=element_text(size=rel(1)),
  strip.text=element_text(size=rel(1), color = 'black'),
  axis.text=element_text(size=rel(0.9)),
  axis.title=element_text(size=rel(1)),
  panel.border= element_blank(),
  strip.background = element_blank(),
  legend.position   ='right',
  plot.title=element_text(size=rel(1), hjust = 0.5),
  text=element_text(size=default_font_size),
  legend.text=element_text(size=rel(1)),
  axis.line.x.bottom = element_blank(),
  axis.line.y.left = element_blank())
theme_set(default_theme)

opts_chunk$set(cache = T)


# n_reps sets the simulation N
```

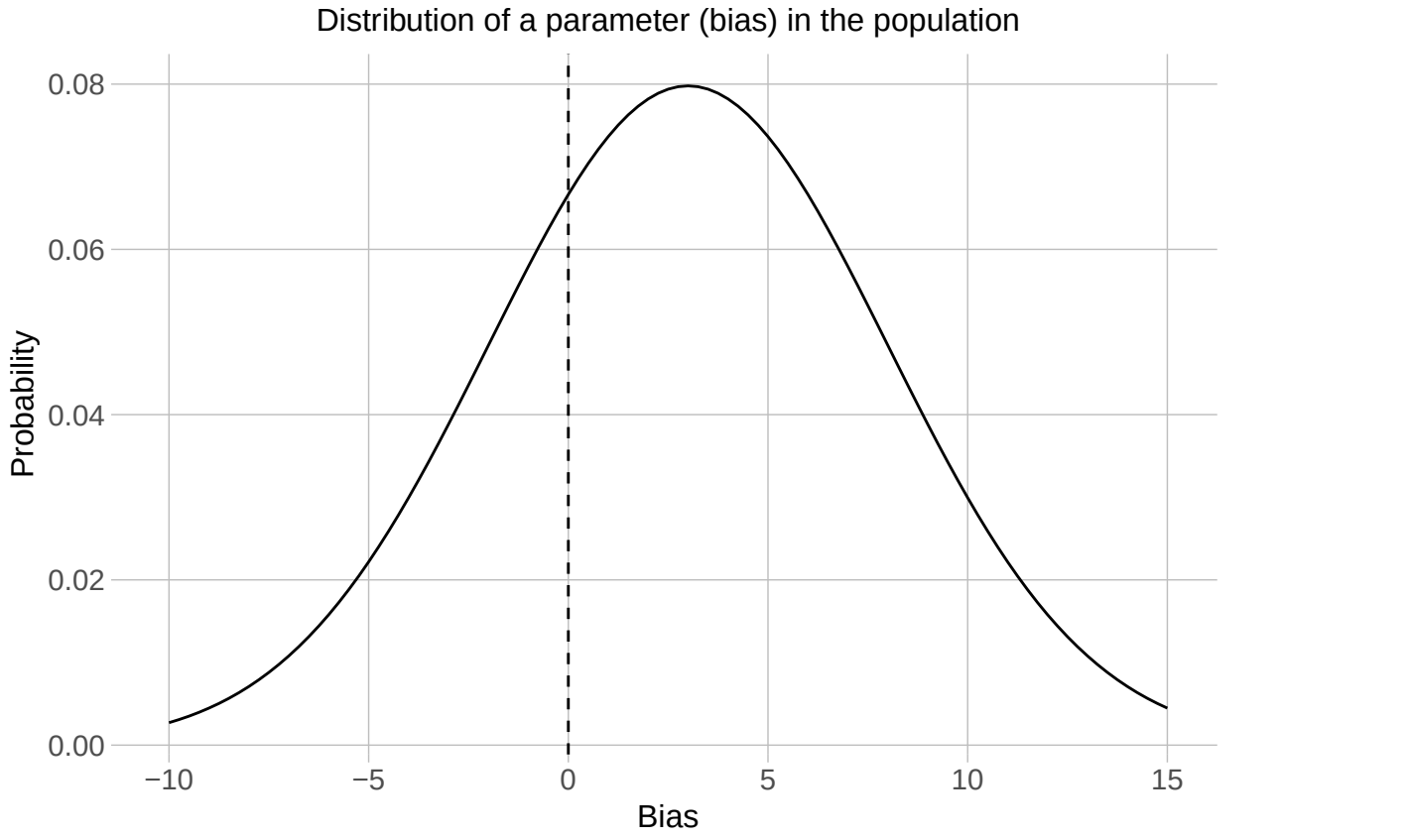
```
max_n_reps = 1000
```

This notebooks aims to demonstrate how to do simulations in R to estimate required sample size and number of trials. First, I'll show how to do simulations for a simple t-test or ANOVA using means and SDs often reported in papers. Then, I'll show how to obtain power estimates by estimating the parameters of within-subject design from existing data and then using it to simulate new datasets.

Simulating data for the one-sample t-test

Let's start with a very simple example. Say, you want to know how many subjects are needed to test a hypothesis that some parameter value (for example, bias in an orientation estimation task) is different from zero with a simple t-test. Here, your data model is that in the population you have a normal distribution of biases with a certain mean and standard deviation.

```
set.seed(512)

mu = 3
sigma = 5
n = 20

p <- ggplot(data.frame(x = seq(-10, 15)), aes(x = x))+
  stat_function(fun = dnorm, args = list(mean = mu, sd = sigma))+
  geom_vline(xintercept = 0, linetype = 2)+
  labs(x = 'Bias', y = 'Probability', title = 'Distribution of a parameter (bias) in the population')

p
```

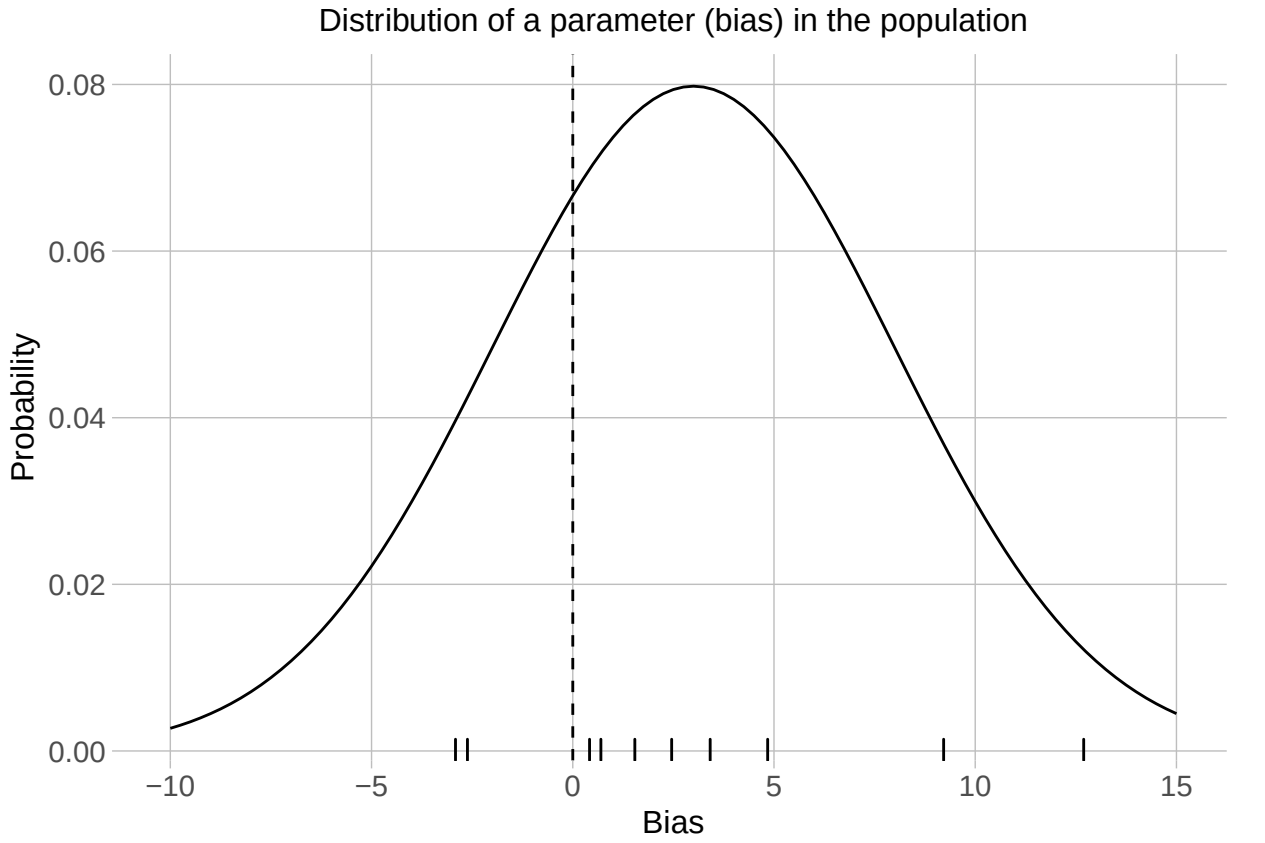


How many subjects do you need to make sure that you will find your effect if the population is what you assume it is? For example, what if you collect data from 10 participants?

```
# generate the sample
sample_data <- rnorm(10, mean = mu, sd = sigma)

# get mean and sd
smean.sd(sample_data)
```

```
##     Mean       SD
## 2.975088 4.911560
```

```
# plot them together with the population distribution
p + geom_rug(data = data.frame(x = sample_data))
```



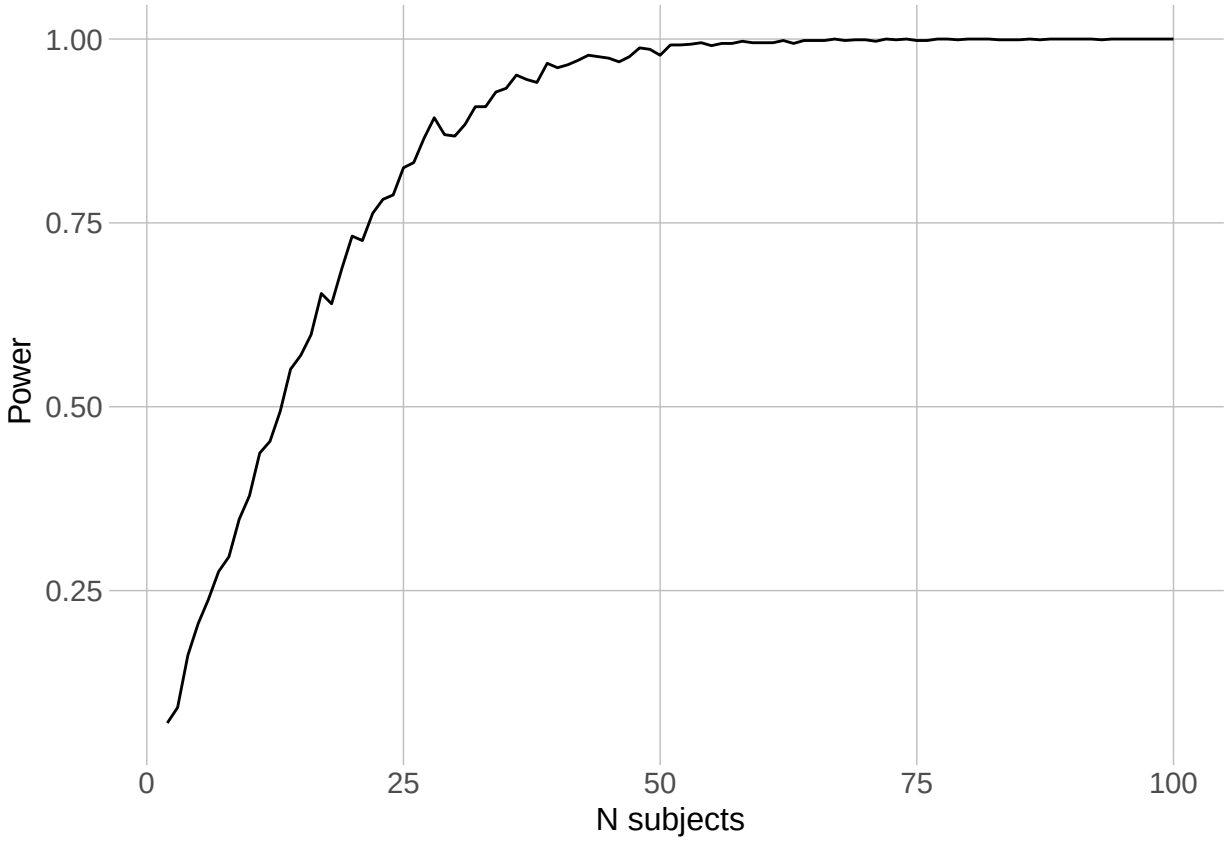

```
# compute t-value and degrees of freedom
t_value <- (mean(sample_data))/(sd(sample_data)/sqrt(length(sample_data)))
df <- length(sample_data)-1

# compute p-value
round.p(2*pt(abs(t_value), df, lower=F))
```

```
## [1] "= .088"
```

For this sample, the effect is not significant. But of course, no one can guarantee that you will get exactly this sample. What you are interested in is the average probability of getting a significant result from your data, also called the "power" of your test.

```
# create a "simulation grid" defining the values for the number of subjects you want to see the power f
```

```
sim_dt <- data.table(expand.grid(n_subjects = 2:100, n_reps = 1:max_n_reps))

# for each replication and each number of subjects, create the sample and compute its mean and SD
# rnorm(...) generates the data for n_subjects
# data.frame(t(...)) transforms the vector output of smean.sd to a one-row data.frame that is easy to c
sim_dt <- sim_dt[, data.frame(t(smean.sd(rnorm(n_subjects, mean = mu, sd = sigma)))), by = .(n_subjects

# then get the t and p like it was done above
sim_dt[, t_value := (Mean)/(SD/sqrt(n_subjects))]
sim_dt[, p_val := 2*pt(abs(t_value), n_subjects-1, lower=F)]

# compute the average probability of getting a significant result
sim_dt_avg<-sim_dt[,.(power = mean(as.numeric(p_val<0.05))), by = n_subjects]

# plot power as a function of the number of subjects
p_power <- ggplot(sim_dt_avg, aes(x = n_subjects, y = power))+
  geom_line(stat='summary', fun.y = mean)+
  labs(y = 'Power', x = 'N subjects')

p_power
```

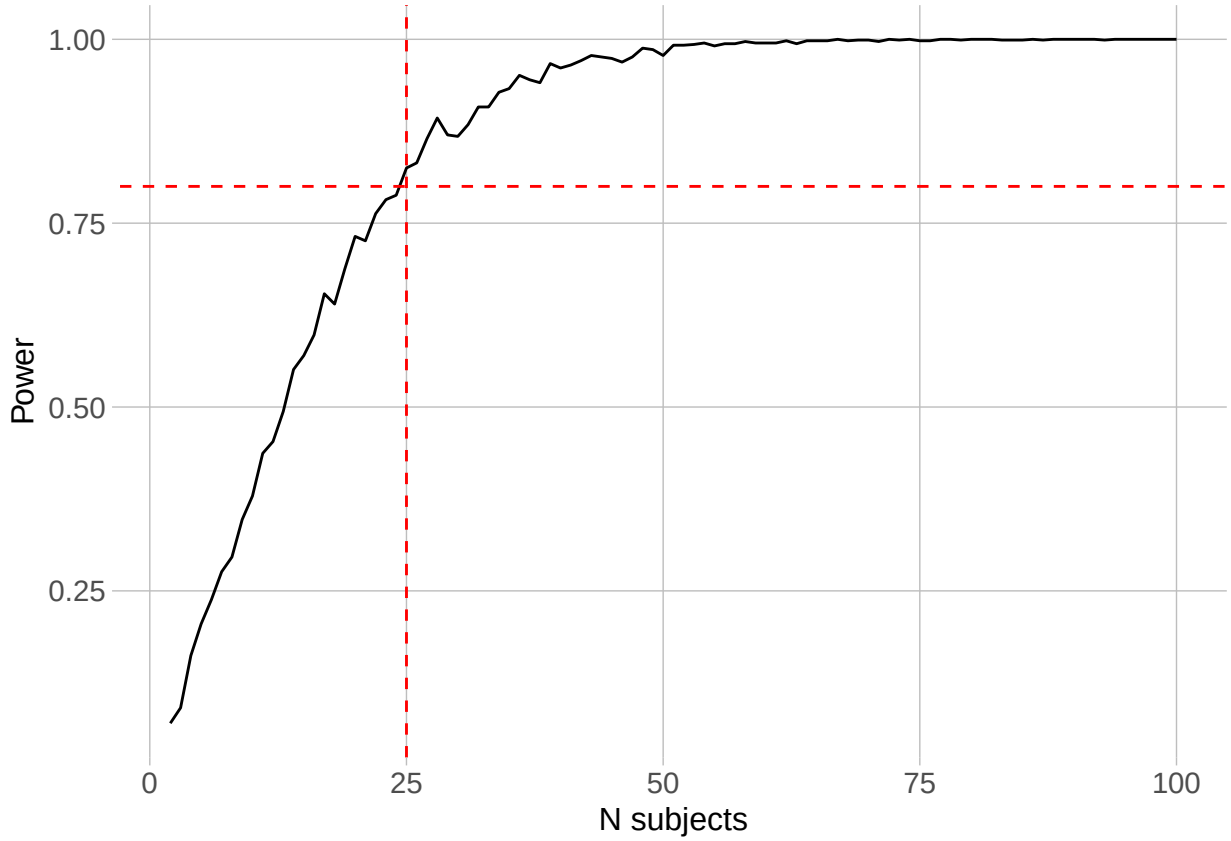


By convention, people often want to know the number of participants to achieve 80% power. Here, in this example, 25 subjects are going to be enough.

```
sim_dt_avg[power>=0.8, min(n_subjects)] # minimal number of subjects needed to achieve 80% power
```

```
## [1] 25
```

```
p_power + geom_hline(yintercept = 0.8, linetype = 2, color = 'red') + geom_vline(xintercept = sim_dt_avg
```

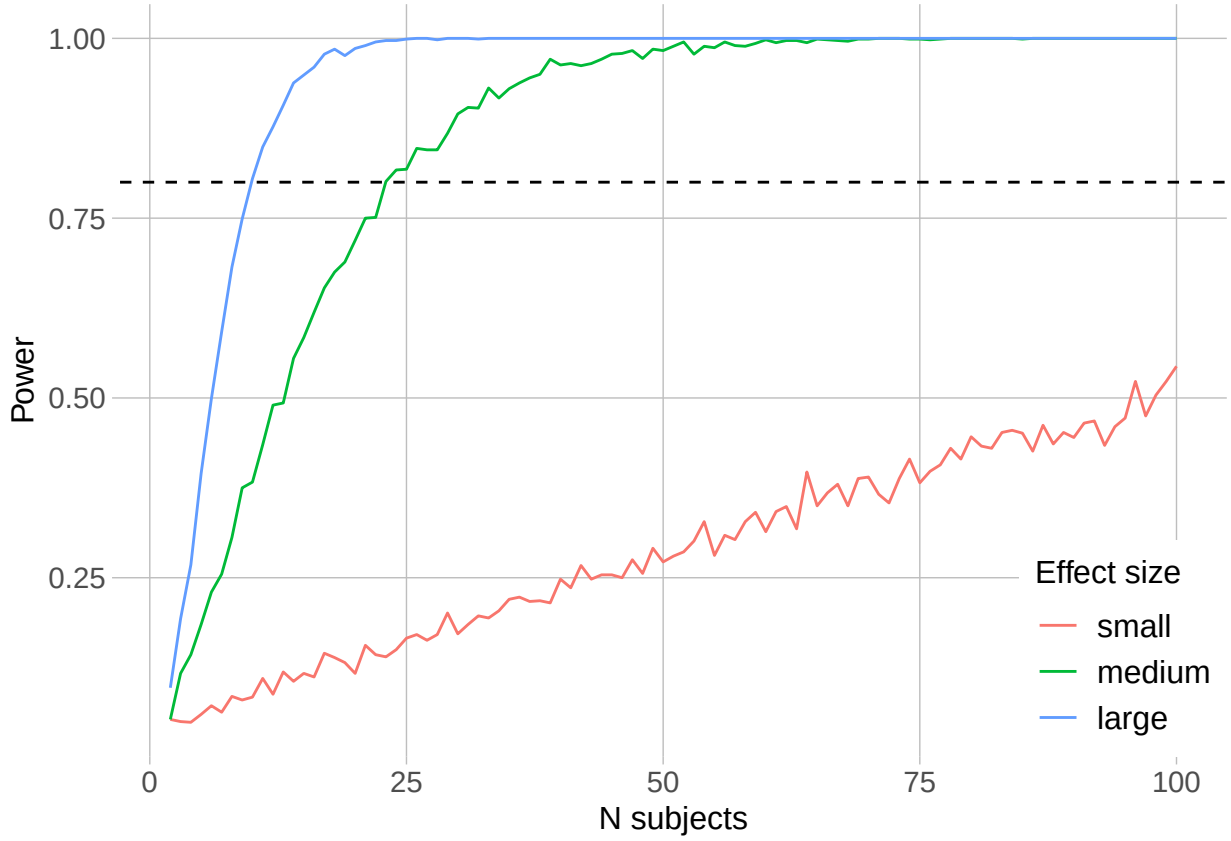


Exercise 1: Simulate and compare the power curves and estimate minimal N for $\sigma = 15$ and $\sigma = 3$.

Of course, power depends on the effect size which (if we use Cohen's d) is just a ratio of mean and SD. So for this example, we used small to medium effect size. We can check how the power differs for different effect sizes.

```
# we are going to check three effect sizes, "small", "medium" and "large"
es_values <- c(0.2, 0.6, 1)

# assuming the same mean (mu = 3), we can get the sigmas for these levels of effect sizes
sigma_values <- mu/es_values

# create a new simulation grid this time varying sigma (SD) of the population distribution
sim_dt <- data.table(expand.grid(n_subjects = 2:100, n_reps = 1:max_n_reps, sigma = sigma_values))

# repeat the power calculations, but this time for different sigma
sim_dt <- sim_dt[, data.frame(t(smean.sd(rnorm(n_subjects, mean = mu, sd = sigma)))), by = .(n_subjects

# then get the t and p
sim_dt[, t_value := (Mean)/(SD/sqrt(n_subjects))]
sim_dt[, p_val:=2*pt(abs(t_value), n_subjects-1, lower=F)]

# compute the average probability of getting a significant result
sim_dt_avg<-sim_dt[,.(power = mean(as.numeric(p_val<0.05))), by = .(n_subjects, sigma)]
```

```
sim_dt_avg[,es_labels := factor(mu/sigma, labels = c('small','medium','large'))]
# plot power as a function of the number of subjects
p_power <- ggplot(sim_dt_avg, aes(x = n_subjects, y = power, color = es_labels))+
  geom_line(stat='summary', fun.y = mean)+
  labs(y = 'Power', x = 'N subjects', color = 'Effect size')+
  theme(legend.position = c(1,0), legend.justification = c(1,0))+
  geom_hline(yintercept = 0.8, linetype = 2, color = 'black')

p_power
```




```
# get the minimal number of subjects needed to achieve 80% power
sim_dt_avg[power>=0.8, min(n_subjects), by = .(es_labels)]
```

```
##    es_labels V1
## 1:    medium 23
## 2:     large 10
```

```
# compare with the results from stats::power.t.test
sapply(sigma_values, function(x) power.t.test(delta = mu, sd = x, power = 0.8, type = 'one.sample' ))
```

```
##             [,1]
## n           198.1513
## delta       3
## sd          15
## sig.level   0.05
## power       0.8
## alternative "two.sided"
## note        NULL
```

```
## method      "One-sample t test power calculation"
##             [,2]
## n           23.79457
## delta       3
## sd          5
## sig.level   0.05
## power       0.8
## alternative "two.sided"
## note        NULL
## method      "One-sample t test power calculation"
##             [,3]
## n           9.937864
## delta       3
## sd          3
## sig.level   0.05
## power       0.8
## alternative "two.sided"
## note        NULL
## method      "One-sample t test power calculation"
```

For a small effect size even one hundred subjects is not enough!

ANOVA

The same logic can be applied to any other test. Let's do it for a between-subject ANOVA just for practice. Say, we have three equally-sized groups with different means and SDs. What is the sample size needed?

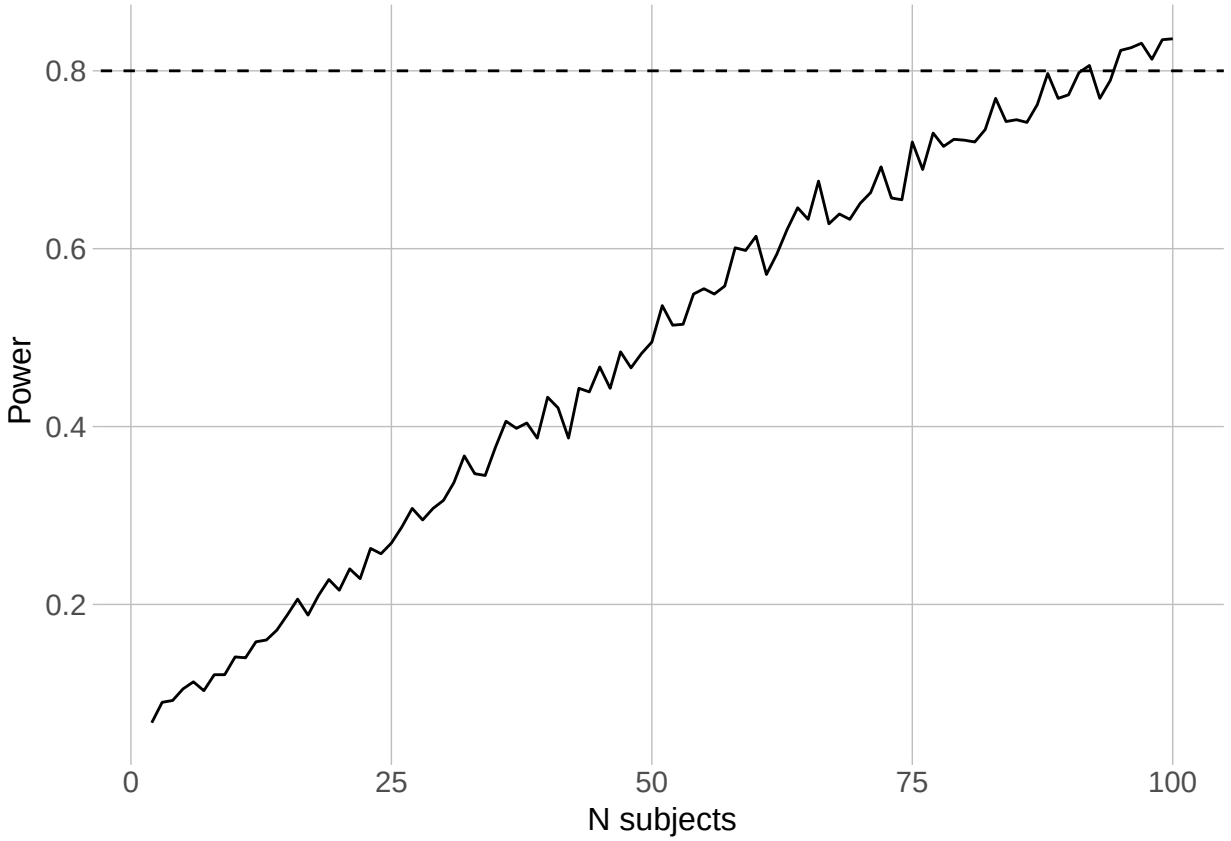
```
group_labels <- c('A','B','C')
means <- c(0, 0.5, 1)
sds <- c(1, 2, 3)

conditions <- data.table(label = group_labels, mu = means, sigma = sds)

# lets try to run ANOVA with N = 20 per group
n_subjects <- 20

# generate the data
data <- conditions[,.( dv = rnorm(n_subjects, mu, sigma)), by = .(label)]
# compute ANOVA
aov_res <- aov(dv~label, conditions[,.( dv = rnorm(n_subjects, mu, sigma)), by = .(label)])
aov_summary <- summary(aov_res)

# for the power simulations, we need to extract p-value. How to do this?
# it's useful to look at the structure of the summary using the str function
str(aov_summary)
```

```
## List of 1
##  $ :Classes 'anova' and 'data.frame':    2 obs. of  5 variables:
##   ..$ Df     : num [1:2] 2 57
##   ..$ Sum Sq : num [1:2] 32.3 345.8
##   ..$ Mean Sq: num [1:2] 16.14 6.07
##   ..$ F value: num [1:2] 2.66 NA
##   ..$ Pr(>F) : num [1:2] 0.0786 NA
##  - attr(*, "class")= chr [1:2] "summary.aov" "listof"
```

```r
# we see that it's a list of length 1 with a data.frame inside
# so to get p_value we first extract the data.frame from a list and then take the first value from `Pr(
aov_summary[[1]]$`Pr(>F)`[1]
```

```
## [1] 0.07858322
```

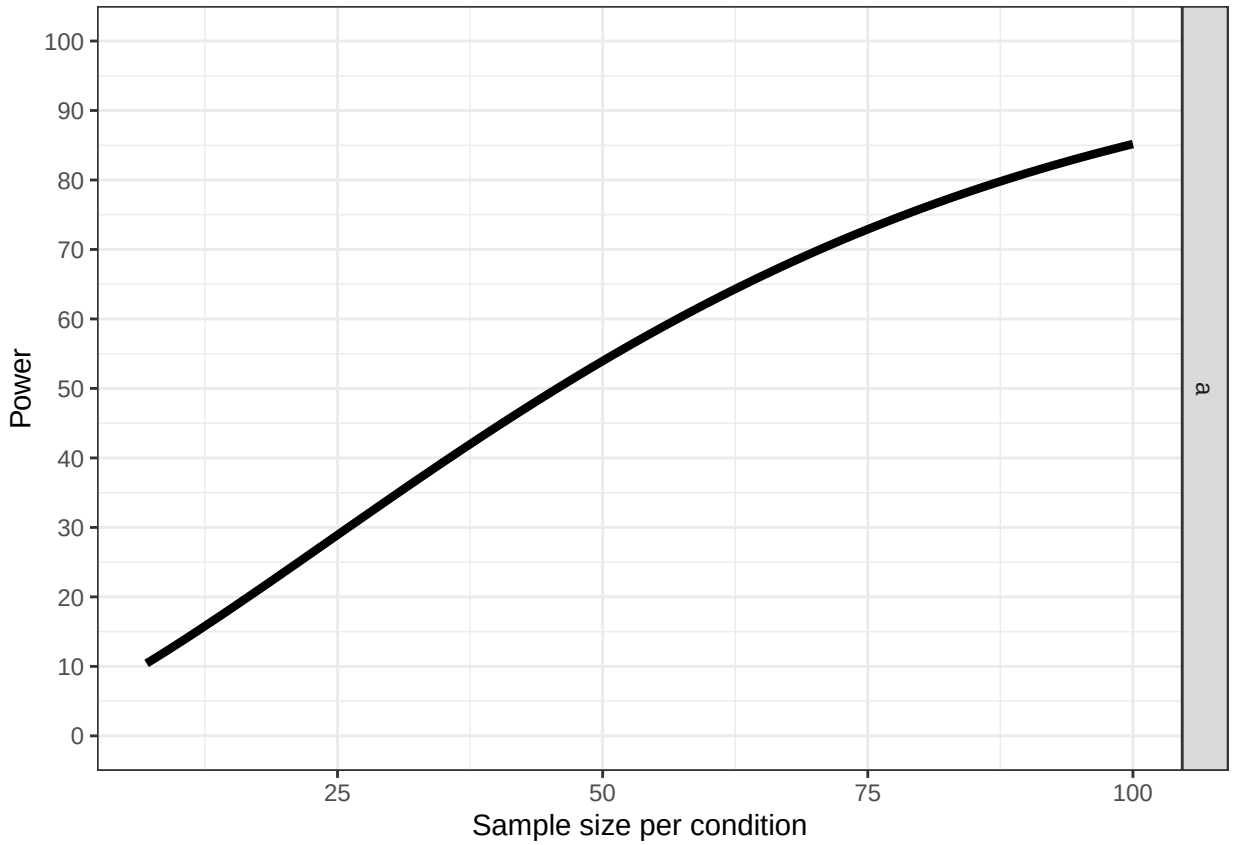
```r
# now, we'll wrap this code in a function for the power calculations
get_sim_anova_p <- function(n_subjects, conditions){
  aov_res <- aov(dv~label, conditions[,.( dv = rnorm(n_subjects, mu, sigma)), by = .(label)])
  summary(aov_res)[[1]]$`Pr(>F)`[1]
}

# create a new simulation grid (the group parameters are fixed, and the n_subjects here is the number o
sim_dt <- data.table(expand.grid(n_subjects = 2:100, n_reps = 1:max_n_reps))

sim_dt <- sim_dt[, p_val:=get_sim_anova_p(n_subjects, conditions ), by = .(n_subjects, n_reps)]

# compute the average probability of getting a significant result
sim_dt_avg<-sim_dt[,.(power = mean(as.numeric(p_val<0.05))), by = .(n_subjects)]

# plot power as a function of the number of subjects
p_power <- ggplot(sim_dt_avg, aes(x = n_subjects, y = power))+
  geom_line(stat='summary', fun.y = mean)+
  labs(y = 'Power', x = 'N subjects')+
  theme(legend.position = c(1,0), legend.justification = c(1,0))+
  geom_hline(yintercept = 0.8, linetype = 2, color = 'black')

p_power
```




```
# get the minimal number of subjects needed to achieve 80% power
sim_dt_avg[power>=0.8, min(n_subjects)]
```

```
## [1] 92
```

```
# compare with the results from Superpower
anovapower_res <- plot_power((ANOVA_design('3b', 30, means, sds)), alpha_level = 0.05, min_n = 7, max_n
```




```
anovapower_res$power_df[anovapower_res$power_df$a>=80,][1,]
```

```
##     n        a
## 82 88 80.03808
```

Exercise 2: Simulate and compare the power curves and estimate minimal N for a two-sample t-test with $\mu = 5$, $\sigma = 10$ and $\mu = 15$, $\sigma = 10$.

```
group_labels <- c('A','B')
means <- c(5, 15)
sds <- c(10, 10)

conditions <- data.table(label = group_labels, mu = means, sigma = sds)

# how to get p-value from t-test? check the structure of t-test object
str(t.test(rnorm(5000)))
```

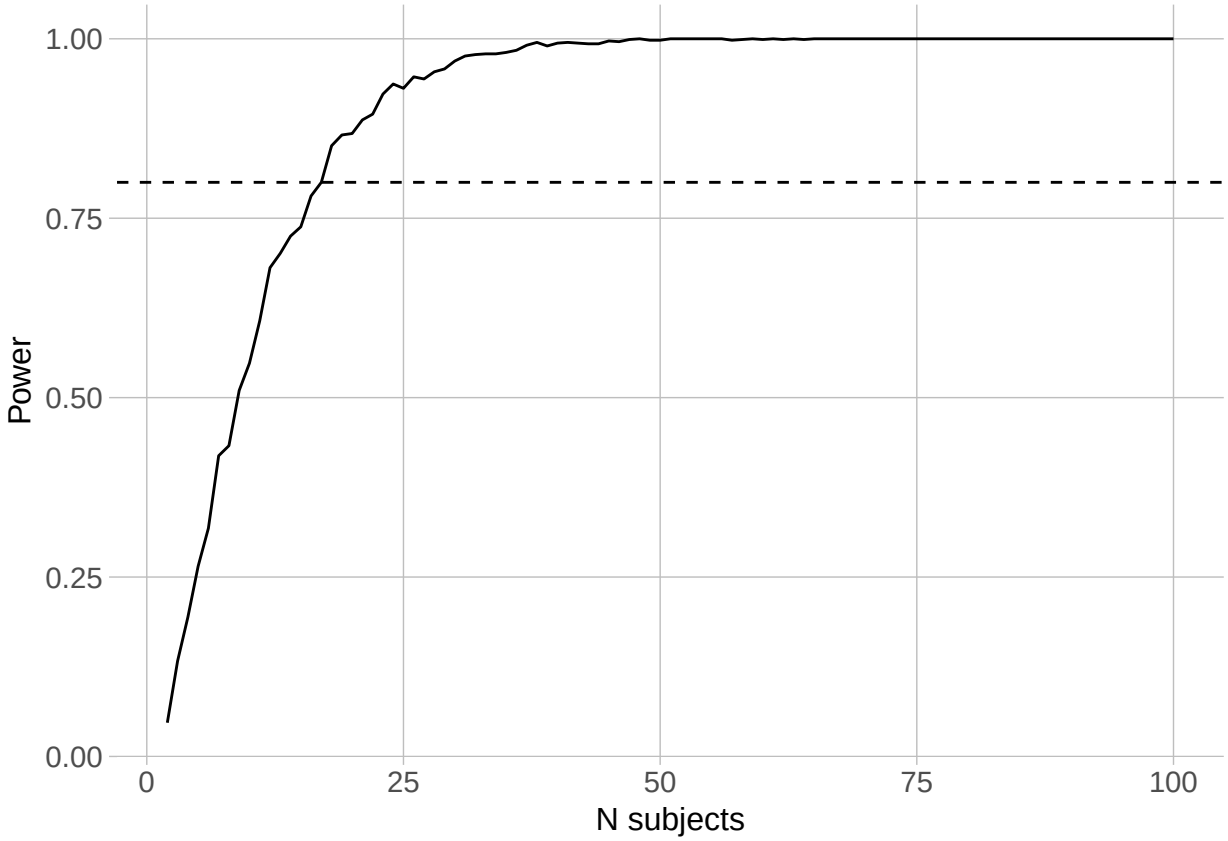
```
## List of 9
##  $ statistic  : Named num 0.511
##   ..- attr(*, "names")= chr "t"
##  $ parameter  : Named num 4999
##   ..- attr(*, "names")= chr "df"
##  $ p.value    : num 0.609
##  $ conf.int   : num [1:2] -0.0207 0.0353
##   ..- attr(*, "conf.level")= num 0.95
##  $ estimate   : Named num 0.00731
##   ..- attr(*, "names")= chr "mean of x"
##  $ null.value : Named num 0
```

```
##   ..- attr(*, "names")= chr "mean"
##  $ alternative: chr "two.sided"
##  $ method     : chr "One Sample t-test"
##  $ data.name  : chr "rnorm(5000)"
##  - attr(*, "class")= chr "htest"
```

```
t.test(rnorm(5000))$p.value
```

```
## [1] 0.6356334
```

```
# now, we'll wrap this code in a function for the power calculations
get_sim_ttest_p <- function(n_subjects, conditions){
  t_res <- t.test(dv~label, conditions[,.( dv = rnorm(n_subjects, mu, sigma)), by = .(label)])
  t_res$p.value
}

# create a new simulation grid (the group parameters are fixed, and the n_subjects here is the number o
sim_dt <- data.table(expand.grid(n_subjects = 2:100, n_reps = 1:max_n_reps))

sim_dt <- sim_dt[, p_val:=get_sim_ttest_p(n_subjects, conditions ), by = .(n_subjects, n_reps)]

# compute the average probability of getting a significant result
sim_dt_avg<-sim_dt[,.(power = mean(as.numeric(p_val<0.05))), by = .(n_subjects)]

# plot power as a function of the number of subjects
p_power <- ggplot(sim_dt_avg, aes(x = n_subjects, y = power))+
  geom_line(stat='summary', fun.y = mean)+
  labs(y = 'Power', x = 'N subjects')+
  theme(legend.position = c(1,0), legend.justification = c(1,0))+
  geom_hline(yintercept = 0.8, linetype = 2, color = 'black')

p_power
```




```
# get the minimal number of subjects needed to achieve 80% power
sim_dt_avg[power>=0.8, min(n_subjects)]
```

```
## [1] 17
```

```
# compare with the results from pwr::pwr.t.test
pwr.t.test(d = (means[2]-means[1])/sqrt(sum(sds^2)/2), power = 0.8, type = 'two.sample' )
```

```
##
##      Two-sample t test power calculation
##
##               n = 16.71472
##               d = 1
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

For simple designs, it is easy to estimate power from summary statistics (e.g., means and variances by group) reported in previous papers. On the other hand, for more complex designs this is not always the case.

Mixed designs

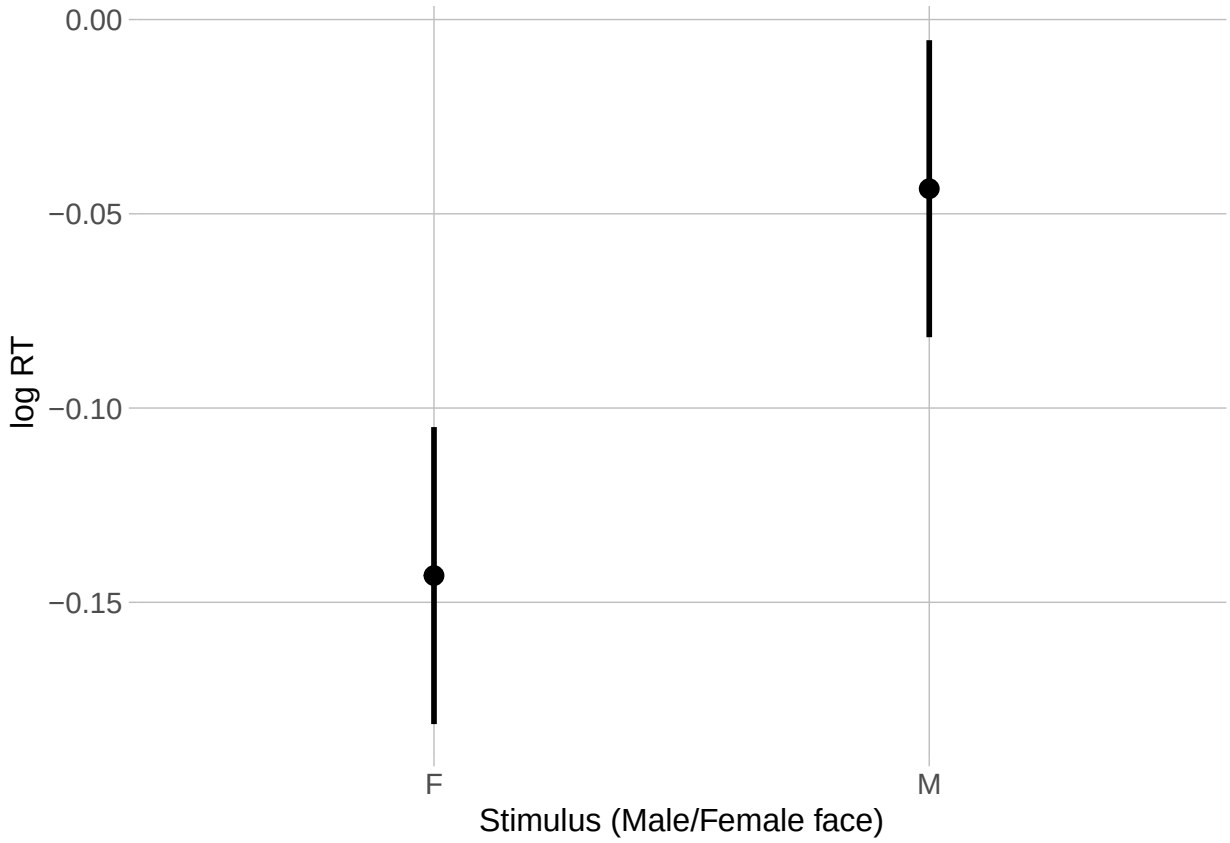
For more complex designs things become funnier. As you probably know, within-subject designs allow accounting for between-subject variability by measuring the same parameter repeatedly for each subject. This

means that if you want to estimate the power of such designs, you need to simulate not only the distribution of parameter values between subjects, but also their distribution within the subjects.

We will use face recognition data from the "apastats" package. From the manual (`? faces`): "This dataset contains data on response times in a face identification task. The faces were presented for a short time, then two faces (old and new) appeared and participants were asked to choose the old one. The data contains the information about gender (both stimuli gender and participants gender) and answer accuracy . . . ".

```
data("faces")

# we'll analyze only the correct answers
faces <- faces[correct==1]

# we'll use log-transformed data to make normal approximation more appropriate
faces[,logRT:=log(answerTime)]
faces[,stim_gender:=factor(stim_gender, levels = c('F','M'))]

# first, we'll just plot the means; plot.pointrange is a ggplot wrapper that allows plotting the means
plot.pointrange(faces, aes(x = stim_gender, y = logRT), wid = 'uid', within_subj = T, do_aggregate = T)
```

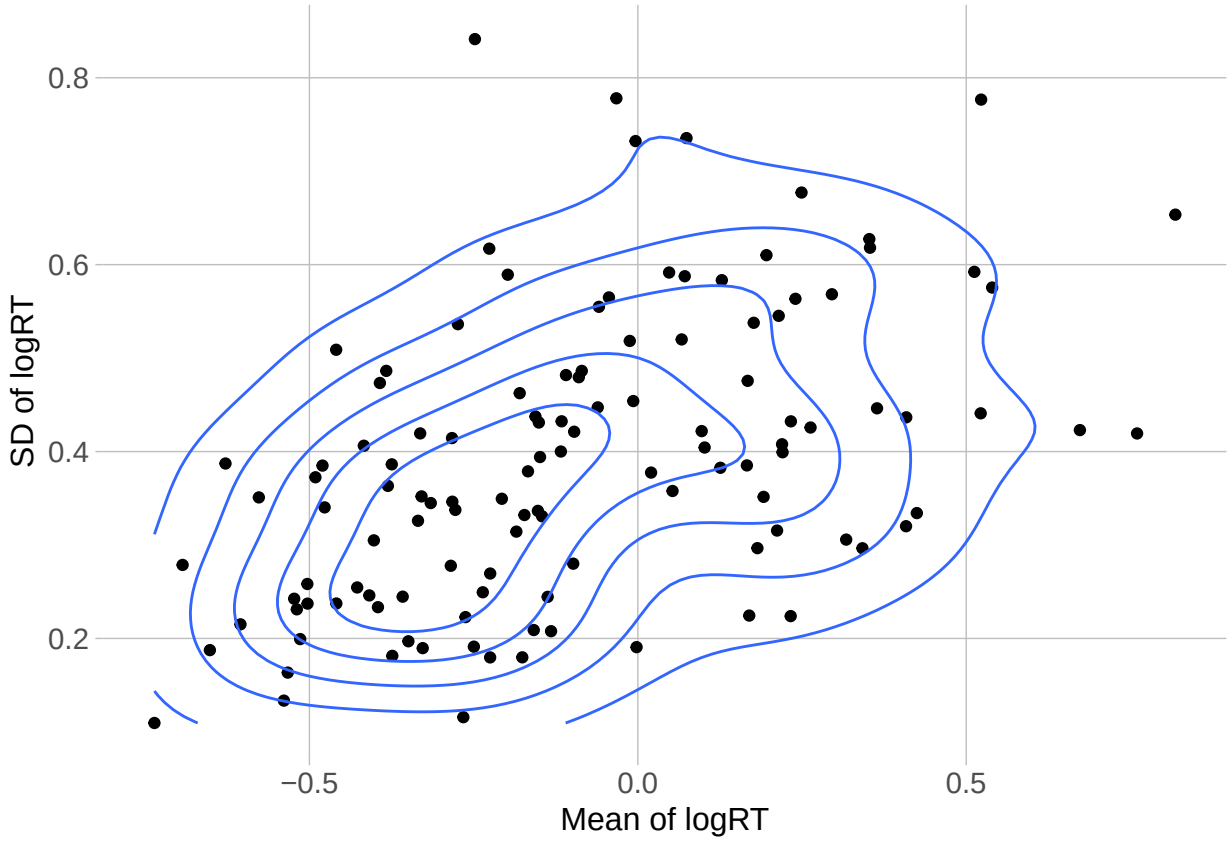
```
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize
```



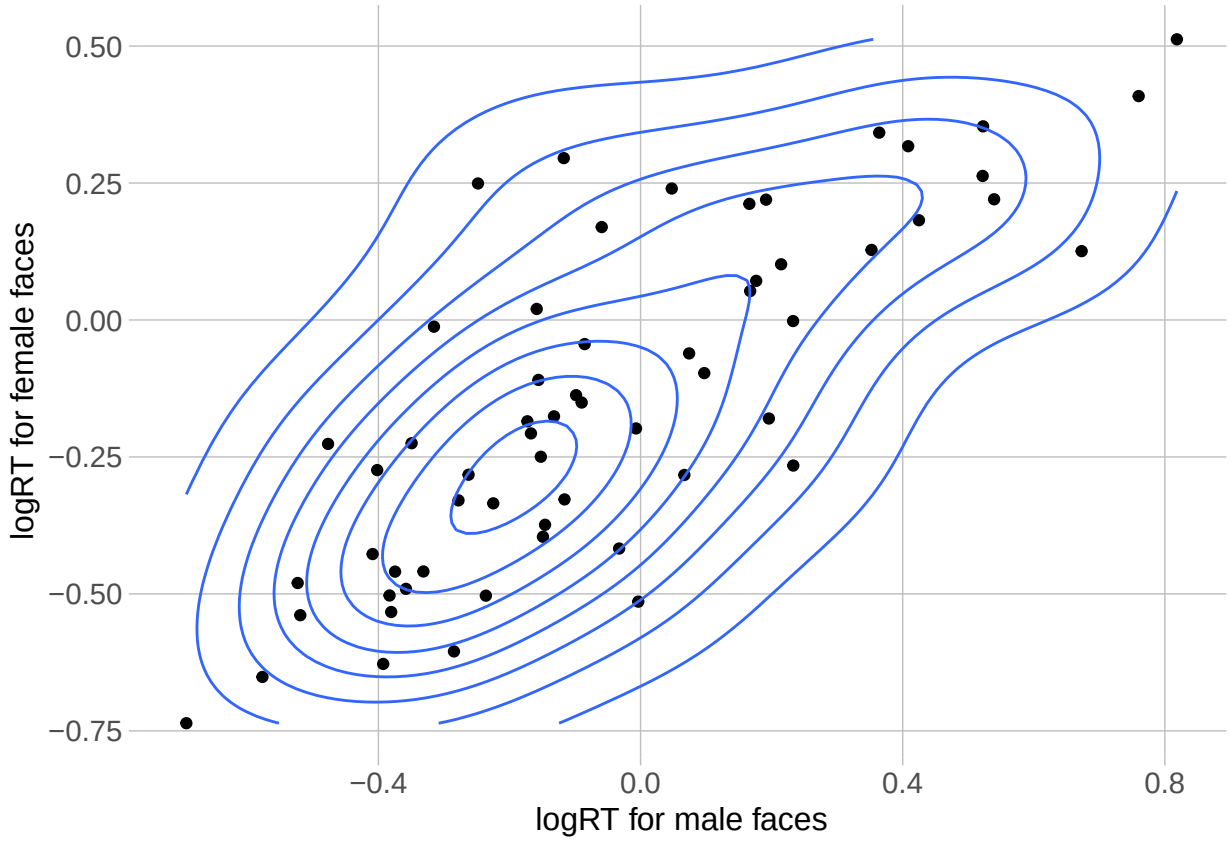
```r
# response times differ (ns.) by condition; to test the significance of the difference we again use jus

faces_aggr <- faces[,.(logRT = mean(logRT)), by = .(uid, stim_gender)]
(t.test(logRT~stim_gender, faces_aggr, paired = T))
```

```
##
##  Paired t-test
##
## data:  logRT by stim_gender
## t = -3.6861, df = 59, p-value = 0.0004971
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1536453 -0.0455256
## sample estimates:
## mean of the differences
##             -0.09958547
```

```r
mean_sd_by_subj <- faces[,data.frame(t(smean.sd(logRT))), keyby = .(uid, stim_gender)]

# reshaping to get mean for each subject in rows
mean_sd_by_subj_re <- dcast(mean_sd_by_subj, uid~stim_gender, value.var = 'Mean')

# plot the results
ggplot(mean_sd_by_subj, aes(x = Mean, y = SD))+geom_point()+stat_density2d()+labs(x = 'Mean of logRT', y
```



```
ggplot(mean_sd_by_subj_re, aes(x = M, y = F))+geom_point()+stat_density2d()+labs(x = 'logRT for male fa
```



As you can see, the parameters of the RT distribution by subject are correlated. To simulate the distribution of these parameters, we will need a covariance matrix. A covariance matrix contains variances of the distribution on the diagonal and covariances off the diagonal.

```
# means
mean_est <- mean_sd_by_subj[,mean(Mean), keyby = stim_gender]$V1

# covariance matrix
cov_mat <- var(mean_sd_by_subj_re[,.(F, M)])

cov_mat
```

```
##            F          M
## F 0.09722490 0.08575558
## M 0.08575558 0.11807971
```

```
# generate multivariate normal data and check its covariance and means
sim_data <- data.frame(mvrnorm(n = 1000, mean_est, cov_mat))
var(sim_data)
```

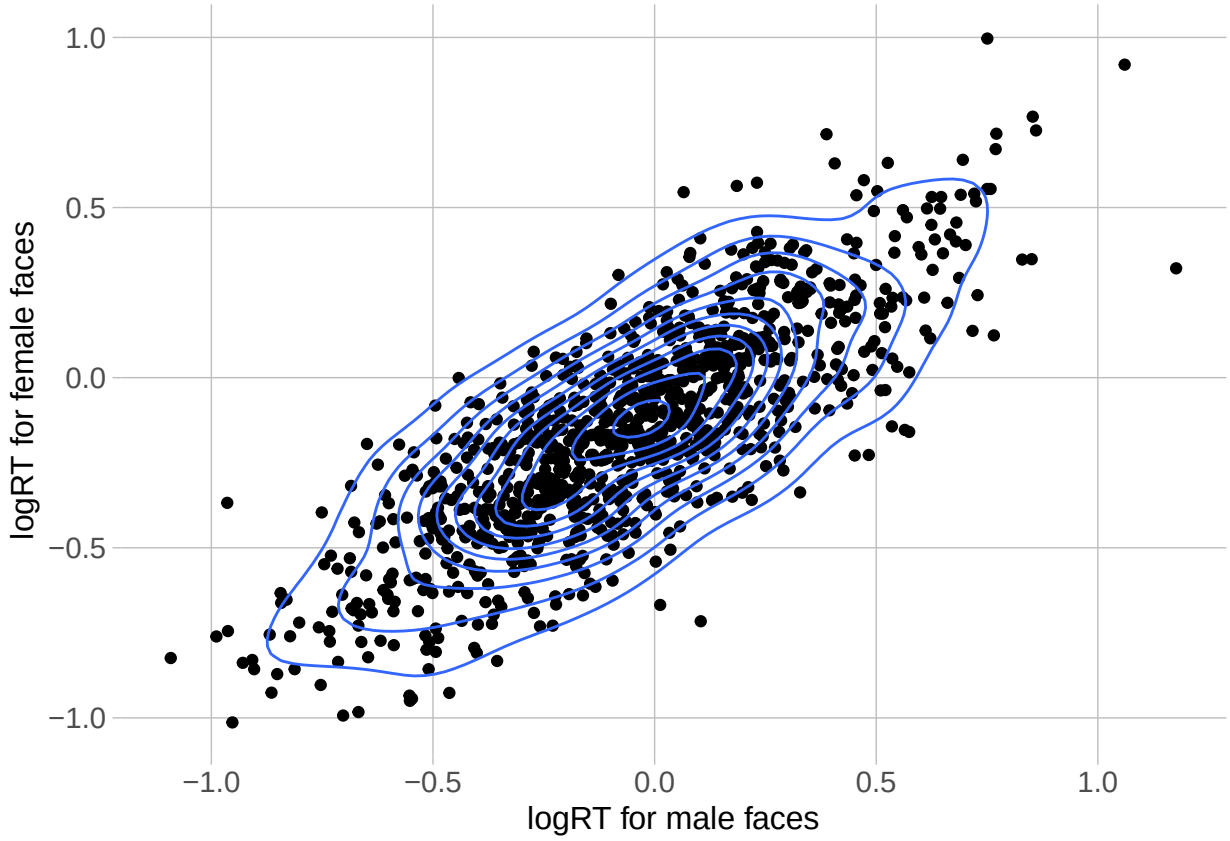
```
##            F          M
## F 0.09898578 0.08672101
## M 0.08672101 0.11725607
```

```
colMeans(sim_data)
```

```
##           F           M
```

```
## -0.15428201 -0.05563457
```

```
# plot the simulated data
ggplot(sim_data, aes(x = M, y = F))+geom_point()+stat_density2d()+labs(x = 'logRT for male faces', y ='
```



For simplicity, we will assume that SDs are the same for all subjects (which they aren't). Otherwise we'll have to make a 4D distribution accounting for covariances in means and SDs in each condition and/or define SD as a function of the mean.

```
# now we again create a function that will do the simulation for a given number of subjects & trials, c

avg_sd <- mean_sd_by_subj[,mean(SD)]

# note that the design is not balanced
faces[,.N, by =.(uid, stim_gender)][,mean(N),by=.(stim_gender)]
```

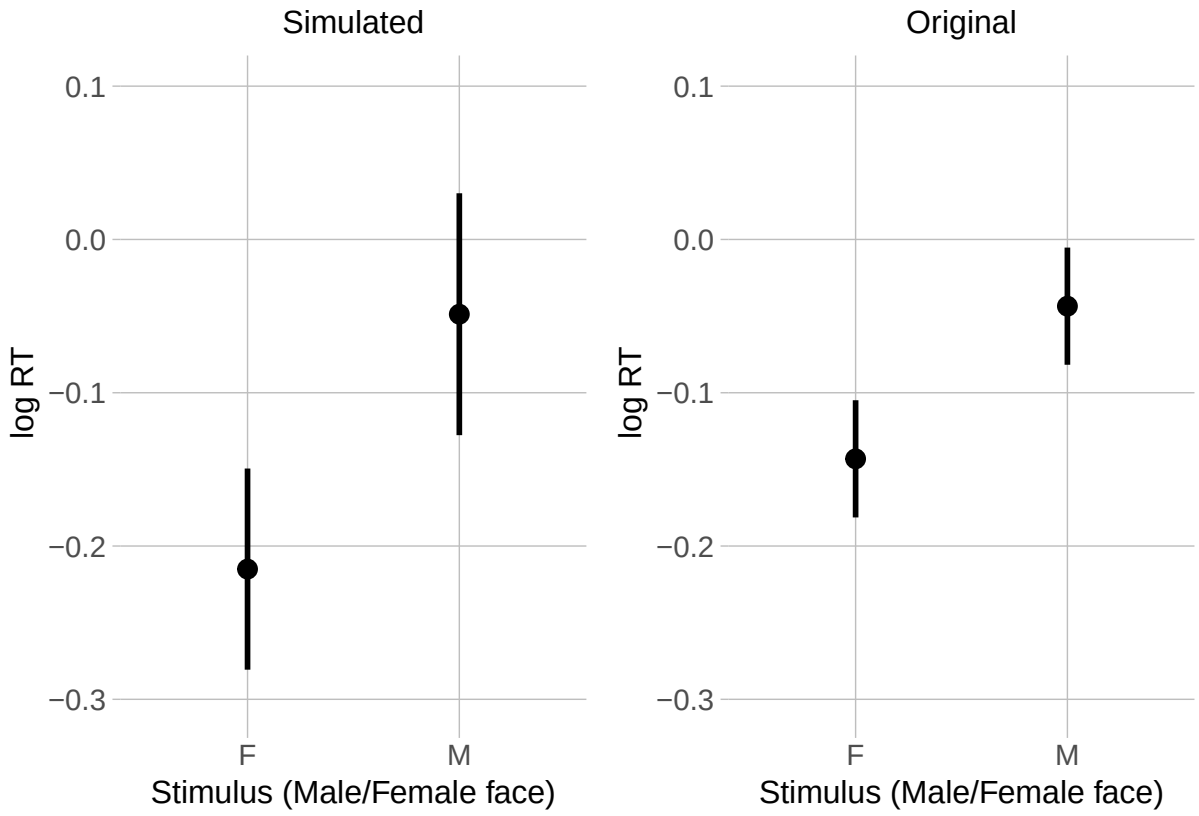
```
##    stim_gender       V1
## 1:           M 20.11667
## 2:           F  9.55000
```

```
# accordingly, we'll use uneven number of trials per condition
n_trials <- round(faces[,.N, by =.(uid, stim_gender)][,mean(N),by=.(stim_gender)]$V1)

simulate_mix <- function (n_subjects, n_trials, mean_est, cov_mat, avg_sd){
  # generate means by subject
  sim_data <- data.table(mvrnorm(n = n_subjects, mean_est, cov_mat))
  # create subject IDs
  sim_data[,uid:=1:.N]
  # for each condition, generate trials
```

```
  sim_data[,rbind(
    data.frame(stim_gender = 'M', logRT = rnorm(n_trials[1], M, avg_sd)),
    data.frame(stim_gender = 'F', logRT = rnorm(n_trials[2], F, avg_sd))), by = .(uid)]
}


# example: simulate data with the original number of trials and subjects
ex_sim <- simulate_mix(n_subjects = 60, n_trials = c(20, 10), mean_est = mean_est, cov_mat = cov_mat, av
ex_sim[,.N, by = .(uid, stim_gender)]
```

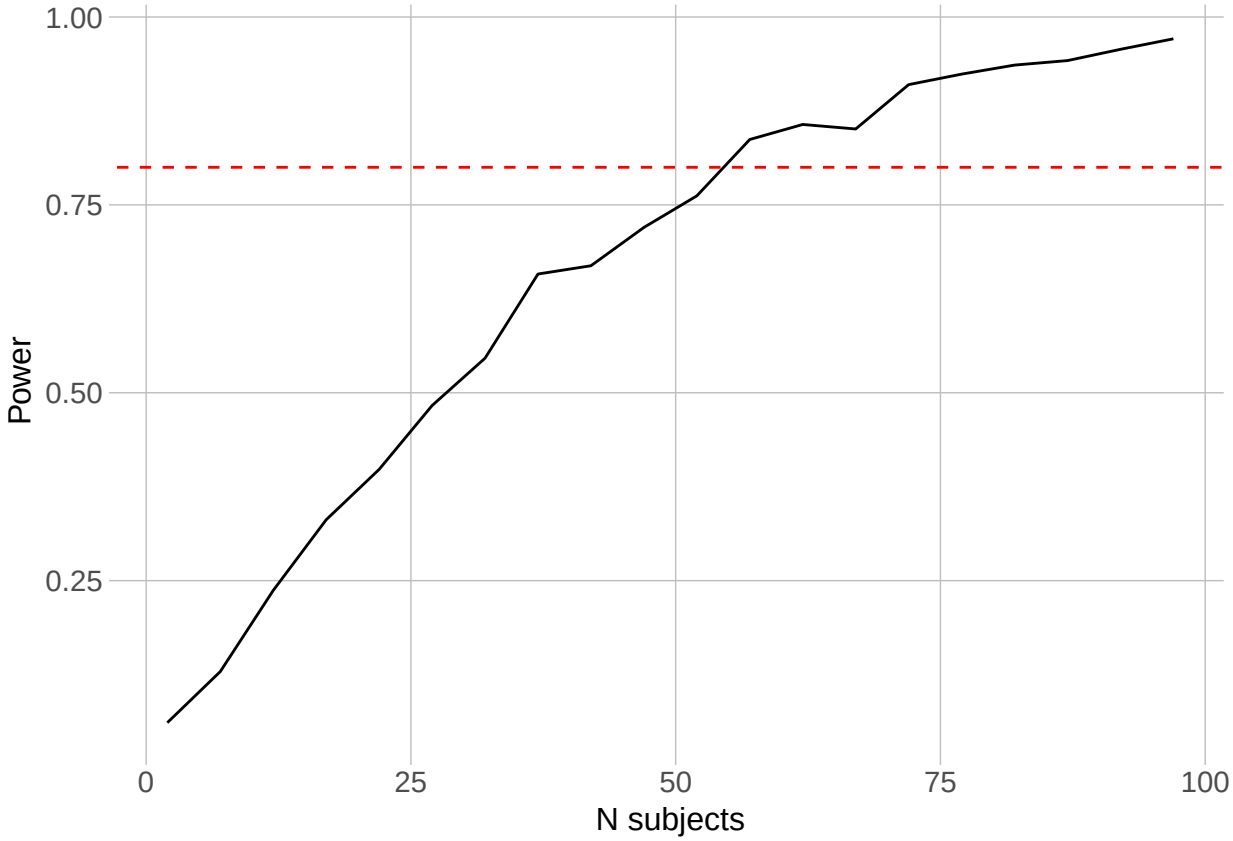
```
##      uid stim_gender  N
##   1:   1           M 20
##   2:   1           F 10
##   3:   2           M 20
##   4:   2           F 10
##   5:   3           M 20
##  ---
## 116:  58           F 10
## 117:  59           M 20
## 118:  59           F 10
## 119:  60           M 20
## 120:  60           F 10
```

```
ex_sim[,stim_gender:=factor(stim_gender, levels = c('F','M'))]

# compare original and simulated data
p1 <- plot.pointrange(ex_sim, aes(x = stim_gender, y = logRT), wid = 'uid', do_aggregate = T) + labs(x =
p2 <- plot.pointrange(faces, aes(x = stim_gender, y = logRT), wid = 'uid', within_subj = T, do_aggregate

(p1+p2)*coord_cartesian(ylim = c(-0.3, 0.1))
```



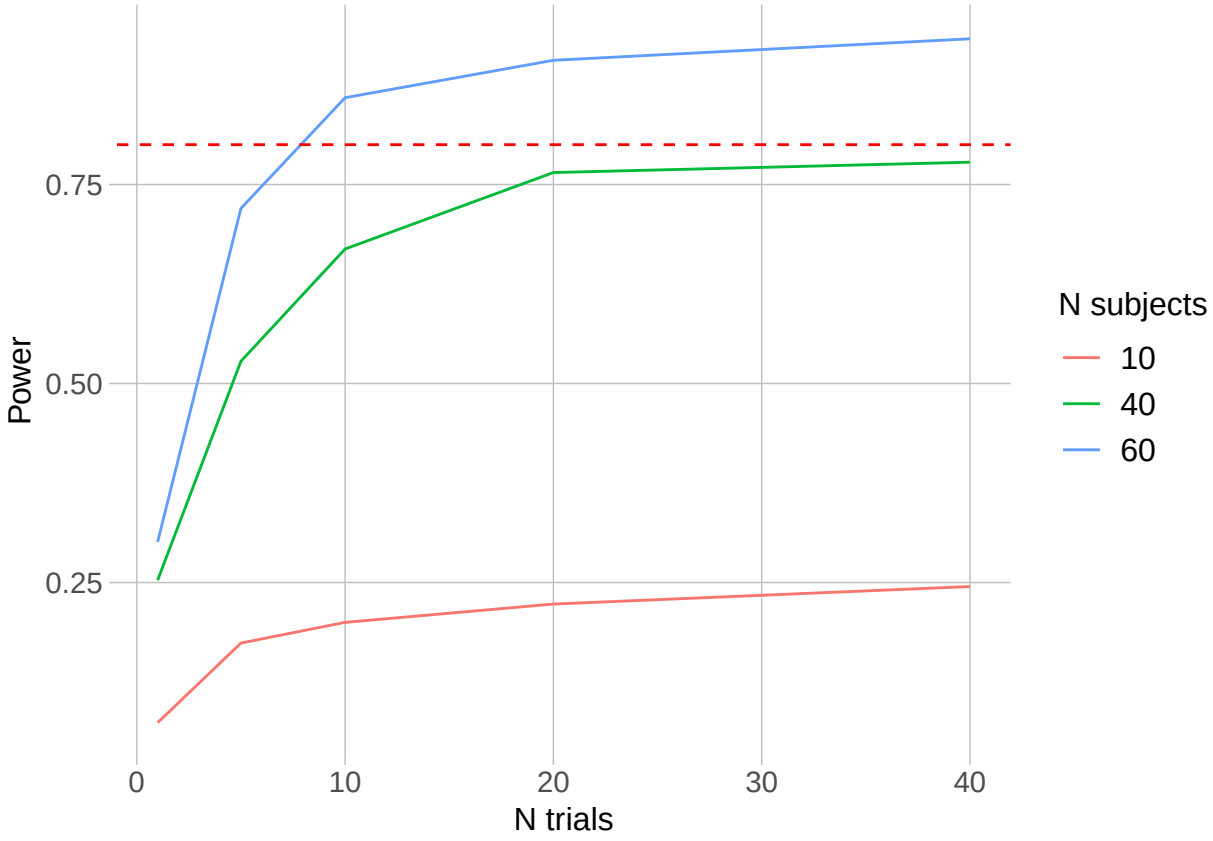
The original and the simulated data should look relatively similar (note that we have made simplifications when approximating the data model so they won't be exactly similar; besides, it is simulated data).

```
# to make things easier, we will make another function that will aggregate the output of simulate_mix a
simulate_mix_aggr <-  function (n_subjects, n_trials, mean_est, cov_mat, avg_sd){
  sim_data <- simulate_mix(n_subjects = n_subjects, n_trials = n_trials, mean_est = mean_est, cov_mat =

  sim_data_aggr <- sim_data[,.(logRT = mean(logRT)), by = .(uid, stim_gender)]
  (t.test(logRT~stim_gender, sim_data_aggr, paired = T))$p.value
}

# now we will simulate data for different numbers of subjects like we did before
# to speed up simulations, we will generate n_subjects in steps of five

sim_dt <- data.table(expand.grid(n_subjects = seq(2,100,5), n_reps = 1:max_n_reps))

# compute p-value for each replication and each sample size
sim_dt[, p_val:= simulate_mix_aggr(n_subjects, n_trials, mean_est, cov_mat, avg_sd), by = .(n_subjects,

# compute the average probability of getting a significant result
sim_dt_avg<-sim_dt[,.(power = mean(as.numeric(p_val<0.05))), by = n_subjects]

ggplot(sim_dt_avg, aes(x = n_subjects, y = power))+
  geom_line(stat='summary', fun.y = mean) +
  geom_hline(yintercept = 0.8, linetype = 2, color = 'red') +
  labs(y = 'Power', x = 'N subjects')
```



An alternative to changing the number of subjects is changing the number of trials. So how do trial numbers affect study power?

```
# we will keep the same proportion of trials by condition, so we only need to set the number of trials
# we will also check three sample sizes

sim_dt_by_trial <- data.table(expand.grid(n_subjects = c(10,40,60), n_reps = 1:max_n_reps, n_trials_f =

# compute p-value for each replication and each sample size and trial number
sim_dt_by_trial[, p_val:= simulate_mix_aggr(n_subjects, n_trials = c(n_trials_f*2, n_trials_f), mean_es

# compute the average probability of getting a significant result
sim_dt_by_trial_avg<-sim_dt_by_trial[,.(power = mean(as.numeric(p_val<0.05))), by = .(n_subjects,n_trial

ggplot(sim_dt_by_trial_avg, aes(x = n_trials_f, color = factor(n_subjects), y = power))+
  geom_line(stat='summary', fun.y = mean)+
  labs(y = 'Power', x = 'N trials', color = 'N subjects')+
   geom_hline(yintercept = 0.8, linetype = 2, color = 'red')
```



Depending on between- and within-subject variability, trial number can be more or less important compared to the sample size. In the current data, power plateaued at about 20 trials in the smaller condition.

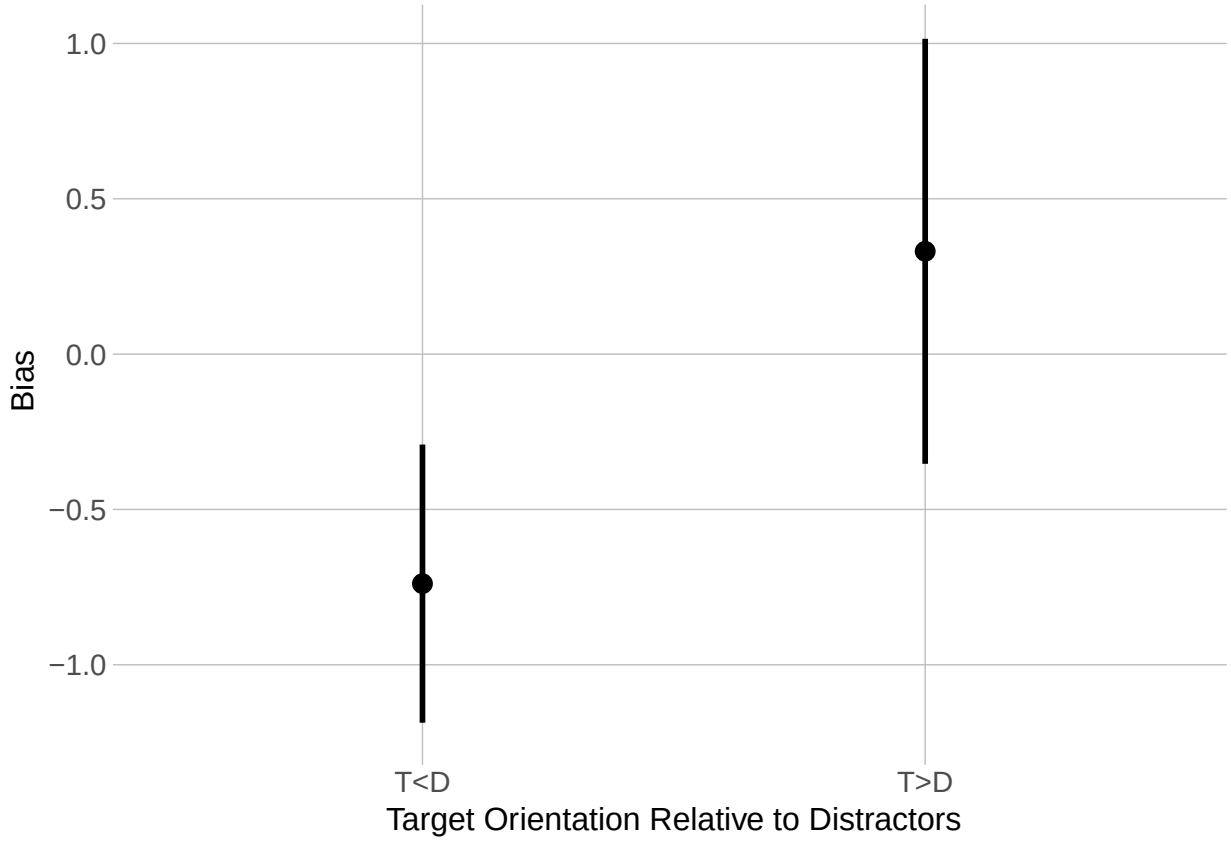
> Exercise 3: Read the data from Raifei et al. (2020) study (`Raifei_2020_simplified.csv`) and estimate the power for 10, 20, 40, and 100 subjects for the average trial N from 100 to 500 in steps of 100 (in both conditions).

Raifei et al. (2020, https://psyarxiv.com/m79nu/) estimated biases in perceived orientation of visual search targets as a function of distractor orientation (clockwise or counterclockwise relative to the target) among other questions

```
data_raifei <- fread(file = 'Raifei_2020_simplified.csv')
# participant - participant ID
# target_distr_rel - a target is oriented clockwise (T>D) or counterclockwise (T<D) relative to distrac
# be_c - error in the adjustment task

str(data_raifei)
```

```
## Classes 'data.table' and 'data.frame':   4160 obs. of  3 variables:
##  $ participant     : chr  "S1" "S1" "S1" "S1" ...
##  $ target_distr_rel: chr  "T>D" "T>D" "T>D" "T<D" ...
##  $ be_c            : num  17.35 20.53 -7.37 7.19 -12.56 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
plot_raifei <- plot.pointrange(data_raifei, aes(x = target_distr_rel, y = be_c), wid = 'participant', d

plot_raifei
```




```
# for tests, you can use ANOVA
ezANOVA(data = data_raifei, dv = be_c, wid = participant, within = target_distr_rel )
```

```
## Warning: Converting "participant" to factor for ANOVA.
```

```
## Warning: Converting "target_distr_rel" to factor for ANOVA.
```

```
## Warning: Collapsing data to cell means. *IF* the requested effects are a subset
## of the full design, you must use the "within_full" argument, else results may be
## inaccurate.
```

```
## $ANOVA
##             Effect DFn DFd       F         p p<.05       ges
## 2 target_distr_rel   1  19 6.72895 0.0178109     * 0.1648493
```

```
# or just a t-test
t.test(be_c~target_distr_rel, data_raifei[,.(be_c = mymean(be_c)), by = .(participant, target_distr_rel
```

```
##
##  Welch Two Sample t-test
##
## data:  be_c by target_distr_rel
## t = -2.7388, df = 32.754, p-value = 0.009896
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.8646446 -0.2748452
## sample estimates:
## mean in group T<D mean in group T>D
##        -0.7388362         0.3309087
```

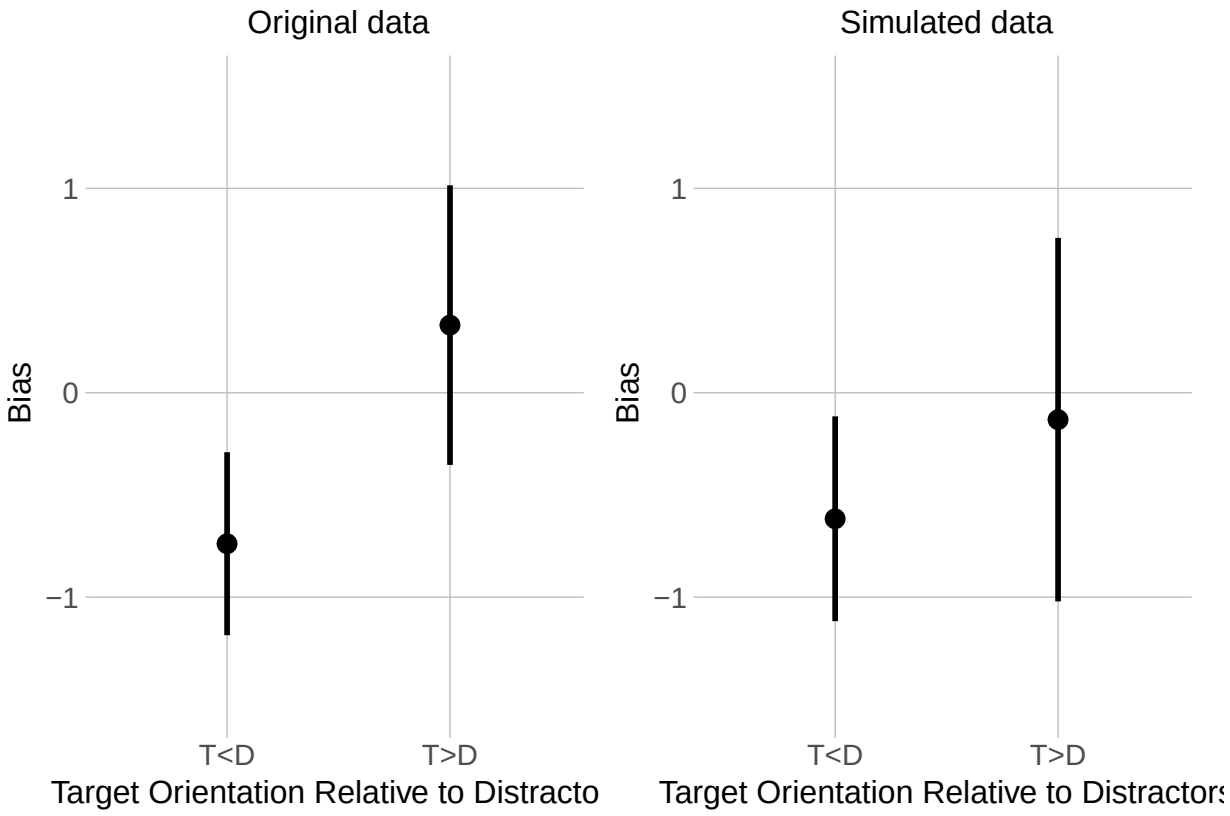
As you can see, when targets are oriented clockwise to distractors (T>D) the error ("bias") is counterclockwise (negative) while when targets are oriented counter-clockwise to distractors (T<D) the error ("bias") is clockwise (positive). The question is, was the number of subjects or trials really enough? Or did the authors just get lucky?

```
# write the function to simulate the data by condition
simulate_mix_raifei <- function (n_subjects, n_trials, mean_est, cov_mat, avg_sd){
  # generate means by subject
  sim_data <- data.table(mvrnorm(n = n_subjects, mean_est, cov_mat))
  # create subject IDs
  sim_data[, participant := 1:.N]
  # for each condition, generate trials
  sim_data[,rbind(
    data.frame(target_distr_rel = 'T<D', be_c = rnorm(n_trials, `T<D`, avg_sd)),
    data.frame(target_distr_rel = 'T>D', be_c = rnorm(n_trials, `T>D`, avg_sd))), by = .(participant)]
}

# get the distribution of means and sds
mean_sd_by_subj <- data_raifei[,data.frame(t(smean.sd(be_c))), keyby = .(participant, target_distr_rel)]

# reshaping to get mean for each subject in rows
mean_sd_by_subj_re <- dcast(mean_sd_by_subj, participant ~ target_distr_rel, value.var = 'Mean')

# get means by conditon
mean_est <- mean_sd_by_subj[, mean(Mean), by = .(target_distr_rel)]$V1

# get covariance matrix
cov_mat <- var(mean_sd_by_subj_re[,.(`T<D`, `T>D`)])

# get average SD
avg_sd <- mean_sd_by_subj[,mymean(SD)]

n_trials <- data_raifei[,.N, keyby = .(participant, target_distr_rel)][,mean(N)]

# example: simulate data with the original number of trials and subjects
ex_sim <- simulate_mix_raifei(n_subjects = 20, n_trials = round(n_trials), mean_est = mean_est, cov_mat

ex_sim[,.N, by = .(participant, target_distr_rel)]
```

```
##     participant target_distr_rel   N
##  1:           1              T<D 104
##  2:           1              T>D 104
##  3:           2              T<D 104
##  4:           2              T>D 104
##  5:           3              T<D 104
##  6:           3              T>D 104
##  7:           4              T<D 104
##  8:           4              T>D 104
##  9:           5              T<D 104
## 10:           5              T>D 104
## 11:           6              T<D 104
## 12:           6              T>D 104
## 13:           7              T<D 104
## 14:           7              T>D 104
## 15:           8              T<D 104
```

```
## 16:           8              T>D 104
## 17:           9              T<D 104
## 18:           9              T>D 104
## 19:          10              T<D 104
## 20:          10              T>D 104
## 21:          11              T<D 104
## 22:          11              T>D 104
## 23:          12              T<D 104
## 24:          12              T>D 104
## 25:          13              T<D 104
## 26:          13              T>D 104
## 27:          14              T<D 104
## 28:          14              T>D 104
## 29:          15              T<D 104
## 30:          15              T>D 104
## 31:          16              T<D 104
## 32:          16              T>D 104
## 33:          17              T<D 104
## 34:          17              T>D 104
## 35:          18              T<D 104
## 36:          18              T>D 104
## 37:          19              T<D 104
## 38:          19              T>D 104
## 39:          20              T<D 104
## 40:          20              T>D 104
##     participant target_distr_rel   N
```

```
# compare original and simulated data
p_sim <- plot.pointrange(ex_sim, aes(x = target_distr_rel, y = be_c), wid = 'participant', do_aggregate

(plot_raifei+ggtitle('Original data')+p_sim)*coord_cartesian(ylim = c(-1.5, 1.5))
```




```
# create the second function to compute the power
simulate_mix_raifei_aggr <- function (n_subjects, n_trials, mean_est, cov_mat, avg_sd){
  sim_data <- simulate_mix_raifei(n_subjects, n_trials, mean_est, cov_mat, avg_sd)

  # average by condition x subject, compute p value
  sim_data[,.(be_c = mean(be_c)), by = .(participant, target_distr_rel)][,t.test(be_c~target_distr_rel,
}

sim_dt_by_trial_raifei <- data.table(expand.grid(n_subjects = c(10,20,40,100), n_reps = 1:max_n_reps, n
sim_dt_by_trial_raifei
```

```
##        n_subjects n_reps n_trials
##     1:         10      1      100
##     2:         20      1      100
##     3:         40      1      100
##     4:        100      1      100
##     5:         10      2      100
##    ---
## 19996:        100    999      500
## 19997:         10   1000      500
## 19998:         20   1000      500
## 19999:         40   1000      500
## 20000:        100   1000      500
```

```
# compute p-value for each replication and each sample size and trial number
sim_dt_by_trial_raifei[, p_val:= simulate_mix_raifei_aggr(n_subjects, n_trials = n_trials, mean_est, co
```

```
sim_dt_by_trial_raifei_avg<-sim_dt_by_trial_raifei[,.(power = mean(as.numeric(p_val<0.05))), by = .(n_su
```

```
ggplot(sim_dt_by_trial_raifei_avg, aes(x = n_trials, color = factor(n_subjects), y = power))+
  geom_line(stat='summary', fun.y = mean)+
  labs(y = 'Power', x = 'N trials', color = 'N subjects')+
   geom_hline(yintercept = 0.8, linetype = 2, color = 'red')
```